FIGURE 1

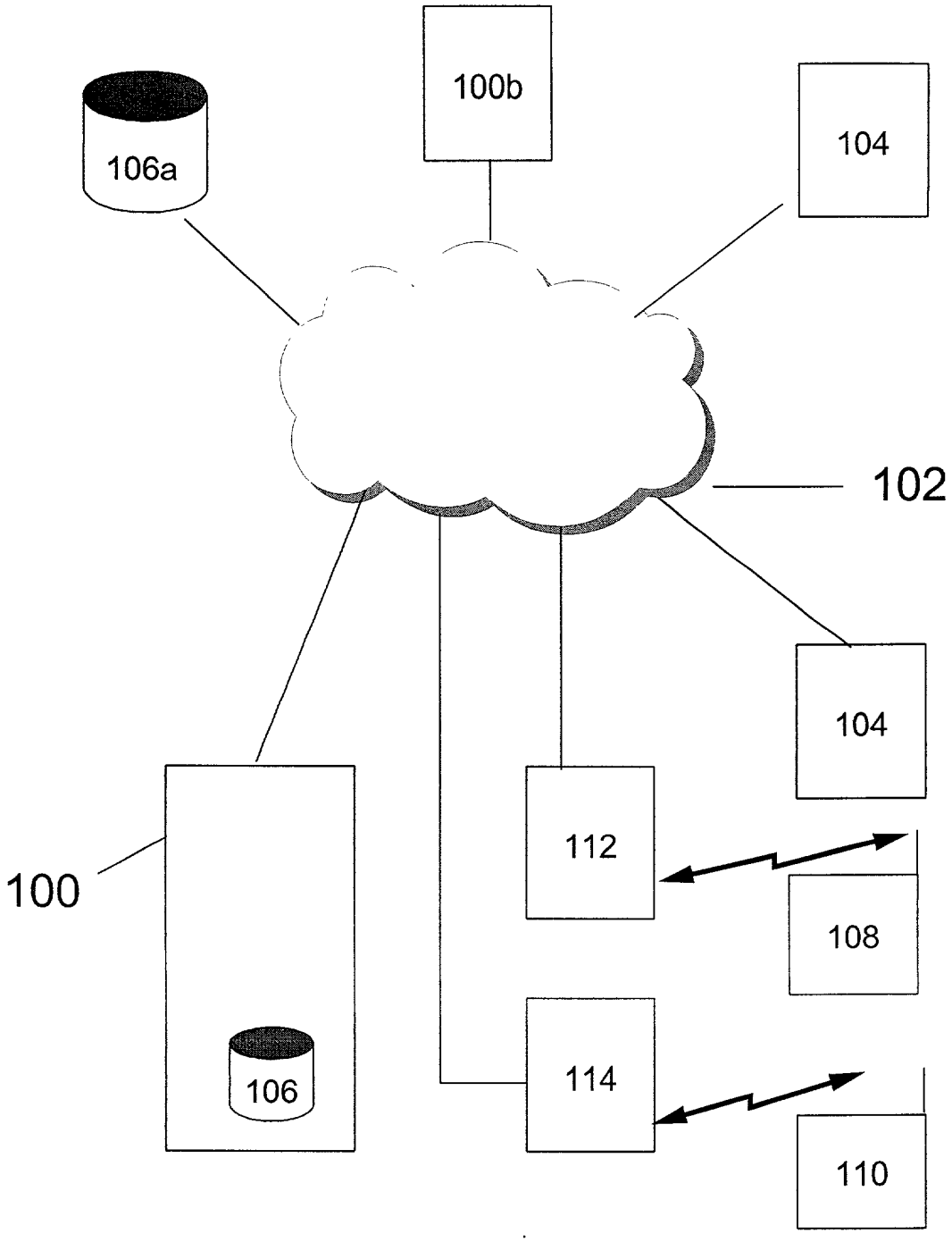


FIGURE 1A

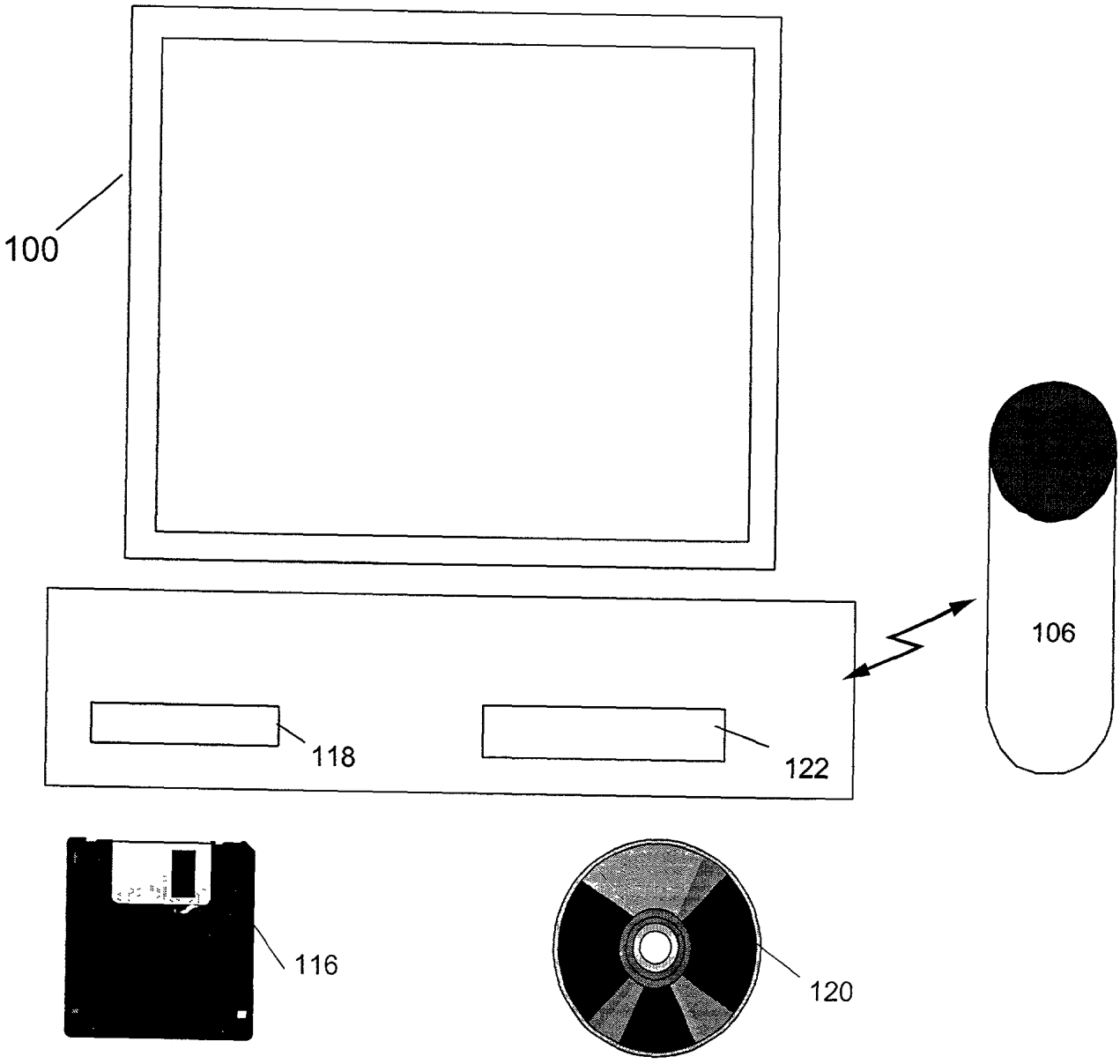


FIGURE 2

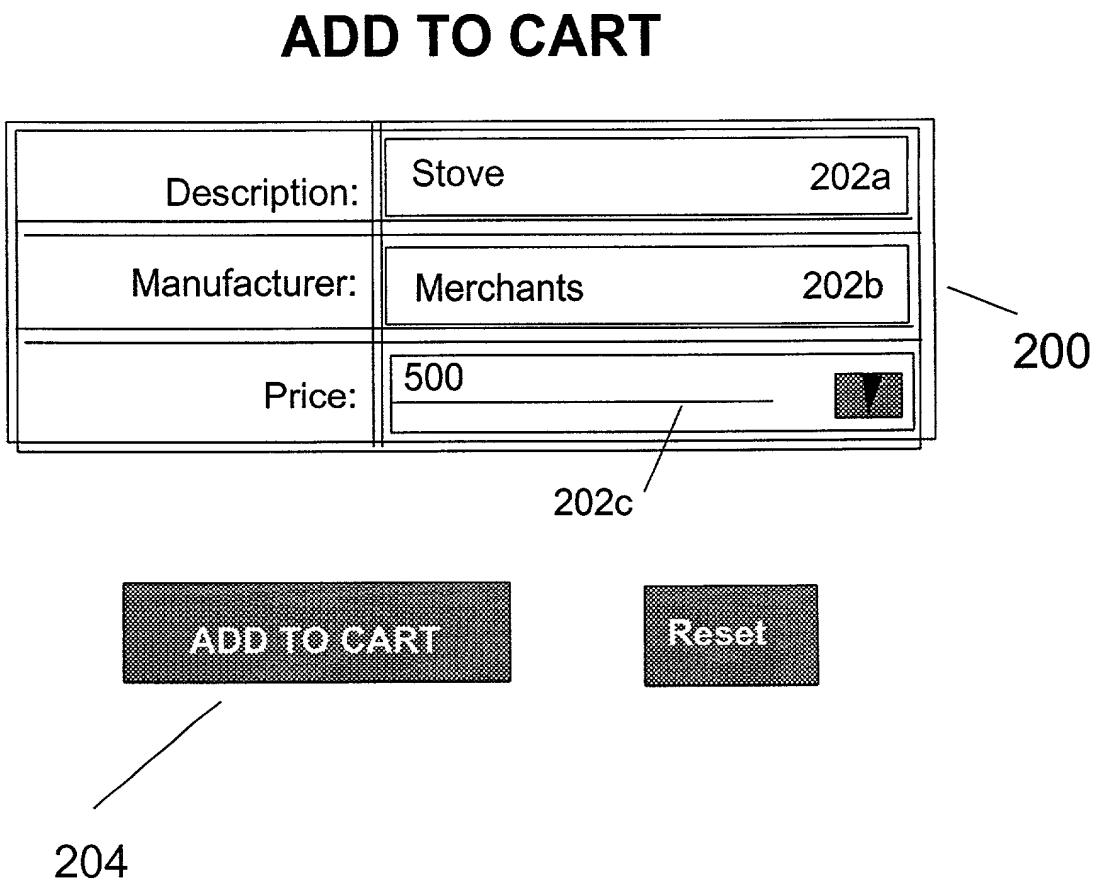

ADD TO CART
Description:
Stove
202a
Manufacturer:
Merchants
202b
Price:
500
200
202c
ADD TO CART
Reset
204

FIGURE 3

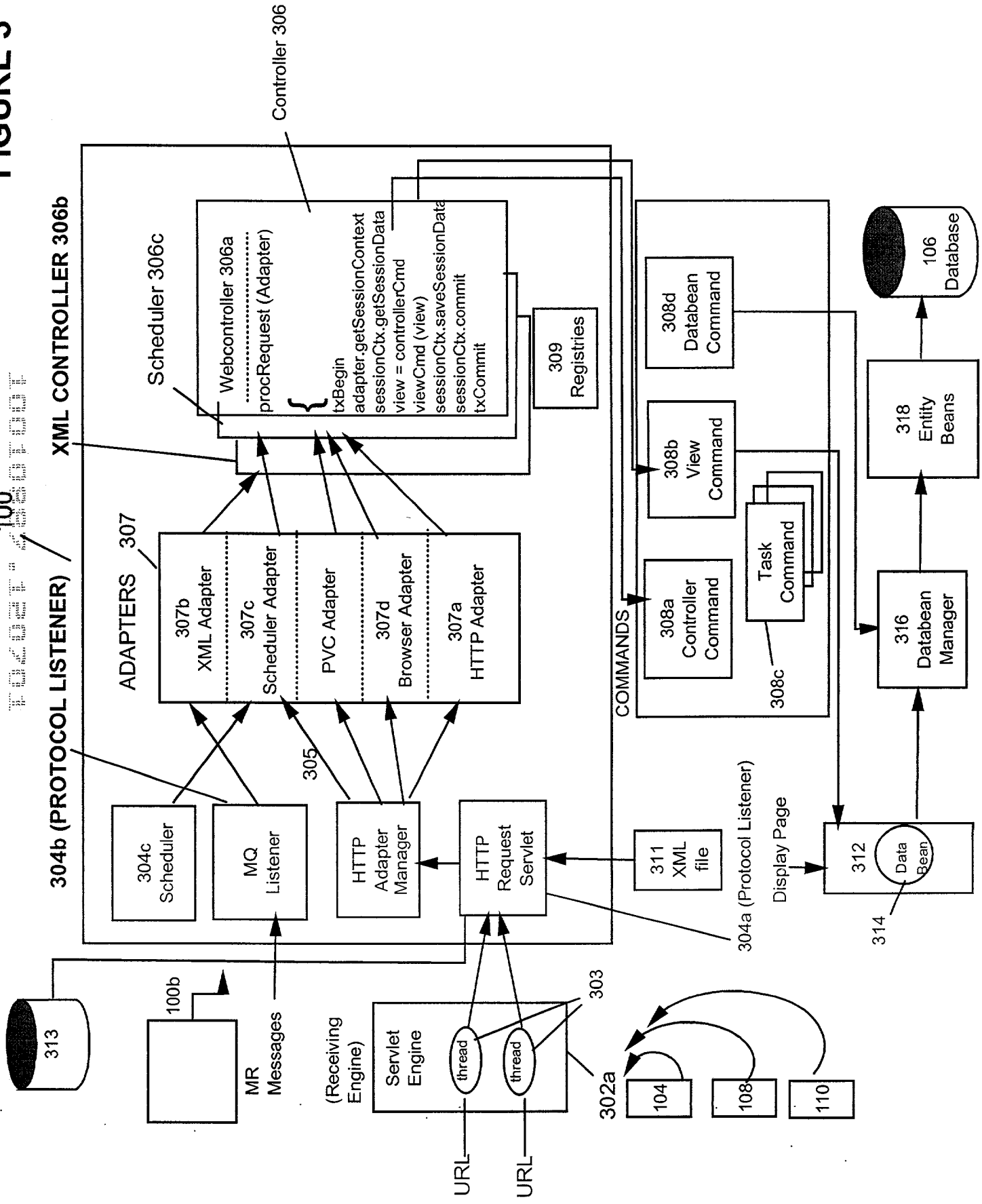

100
XML CONTROLLER 306b
304b (PROTOCOL LISTENER)
Controller 306
Scheduler 306c
Webcontroller 306a
procRequest (Adapter)
txBegin
adapter.getSessionContext
sessionCtx.getSessionData
view = controllerCmd
viewCmd (view)
sessionCtx.saveSessionData
sessionCtx.commit
txCommit
309 Registries
ADAPTERS 307
307b XML Adapter
307c Scheduler Adapter
PVC Adapter
307d Browser Adapter
307a HTTP Adapter
305
304c Scheduler
MQ Listener
HTTP Adapter Manager
HTTP Request Servlet
311 XML file
304a (Protocol Listener)
COMMANDS
308a Controller Command
308b View Command
Task Command
308c
308d Databean Command
316 Databean Manager
318 Entity Beans
106 Database
Display Page
312
Data Bean
314
313
100b
MR Messages
(Receiving Engine)
Servlet Engine
thread
thread
303
302a
104
108
110
URL
URL

FIGURE 3a

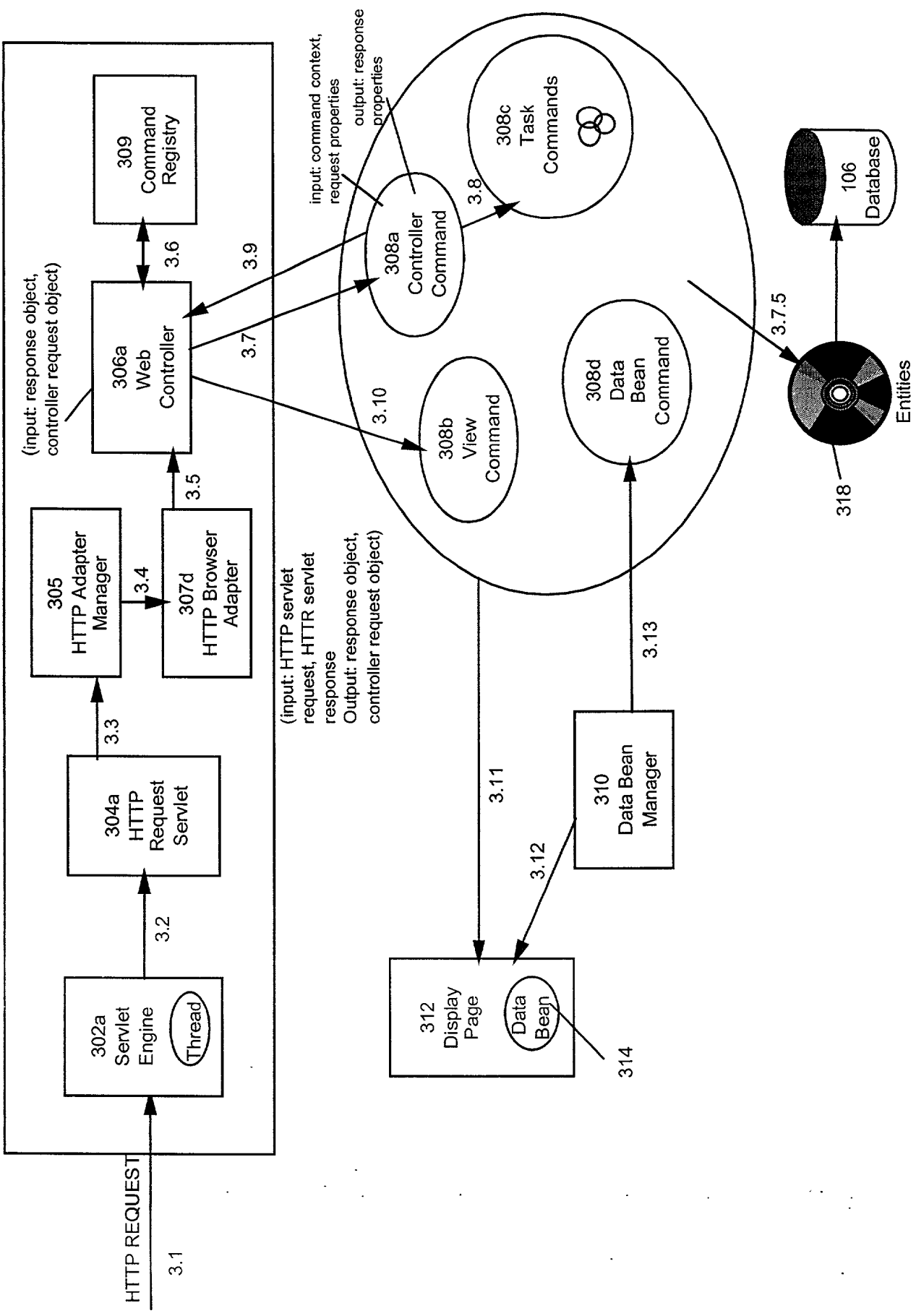

HTTP REQUEST
3.1
302a Servlet Engine
Thread
3.2
304a HTTP Request Servlet
3.3
305 HTTP Adapter Manager
3.4
307d HTTP Browser Adapter
3.5
306a Web Controller
(input: response object, controller request object)
3.6
309 Command Registry
3.7
3.9
(input: HTTP servlet request, HTTR servlet response
Output: response object, controller request object)
input: command context, request properties
output: response properties
308a Controller Command
3.8
308c Task Commands
3.10
308b View Command
308d Data Bean Command
3.7.5
106 Database
318
Entities
3.11
3.12
3.13
310 Data Bean Manager
312 Display Page
Data Bean
314

Fig. 4A

```
//
// Http Request Servlet
//
public class RequestServlet extends HttpServlet {
    .
    .
    .
    void service(HttpServletRequest request, HttpServletResponse response) throws

ServletException, IOException {
            // get an device format adapter that recognizes and handle this request
            // format from a Http Device Format Manager
            // the request can come from a browser or a mobile device or
            // any other source that conforms to the Http protocol

400         HttpAdapter adapter = HttpDeviceFormatManager.getAdapter(request,response);

            // ask the adapter to convert the process the request
            // the adapter convert the request to a RequestObject recognized by the
            // web controller and invoke the processRequest() method on the web
             // controller

402         adapter.processRequest();
}
    .
    .
    .
}
```

Fig. 4B(i)

```
//
// DeviceFormatAdapter
// - defines the basic interface that defines a device format adater
//

interface DeviceFormatAdapter {
        .
        // returns a device format id
        getDeviceFormatId();
        // returns the device type
        getDeviceType();
        // returns a adapter specific session context
        getSessionContext();
        .
}

//
// HttpAdapter
// - defines a Http specific device format adapter
//

interface HttpAdapter extends DeviceFormatAdapter {
        // return a the HttpServletRequest
        getRequest();

        // returns the input parameters
        getRequestProperties();

        // process request
        processRequest();
}
```

Fig. 4B(ii)

```
//
// HttpAdapterBaseImpl
//
abstract class HttpAdapterBaseImpl implements HttpAdapter {

        HttpServletRequest req;
        HttpServletResponse res;
        HttpAdapterBaseImpl(HttpServletRequest req, HttpServletResponse res) {
                // construct new instance of the adapter and initialize it with the request
                // and response
        }

        createRequestObject() {
                // build a RequestObject based on the request information
        }

        processRequest() {
                // convert from HttpServletRequest into / RequestObject
                RequestObject reqobj = createRequestObject();

                // pass request object and response object to web controller
                HttpWebController.processRequest(reqobj, res);
        }

        getRequest() {
                return req;
        }

        TypedProperty getRequestProperties() {
                // extract request properties from request and put in in a TypedProperty
        }
}

//
// HttpBrowserAdapter
//
public class HttpBrowserAdapter extends HttpAdapterBaseImpl {

        .
        SessionContext getSessionContext() {
        // return an Http Browser sepcific session context
        }
}
//
// HttpPVCAdapter
//
public class HttpPVCAdapter extends HttpAdapterBaseImpl implement HttpAdapter {

        .
        SessionContext getSessionContext() {
        // return a PVC sepcific session context
        }
    }
```

Fig. 4C(i)

```
//
// RequestObject - defines the request object that is passed to the web controller
// from any network device
// each adapter can have add adapter specific extension to this
// for example. The Http Adapter adds the HttpServletRequest to this interface
//
interface RequestObject {                                          )
    // return the adapter used to format the incoming request       )
    getDeviceFormatAdapter();                                      )
    // returns the input properties for the command                )
    getRequestProperties();                                        )
    // returns the session context                                 )
    getSessionContext();                                           )
    // sets the adapter used to process this request               )   404
    setDeviceFormatAdapter();                                      )
    // set the input properties                                    )
    setRequestProperties();                                        )
    // sets the session context                                    )
    setSessionContext();                                           )
    // gets the command name                                       )
    get CommandName();                                             )

}
//
// CommandContext - defines the information that can be accessible to the
// command and the web controller to process a command
//
interface CommandContext() {
    .
    // returns the device type                                     )
    getDeviceType();                                               )
    // returns the input properties for the command                )
    getRequestProperties();                                        )
    // returns the store Id                                        )
    getStoreId();                                                  )
    // returns the user id                                         )   406
    getUserId();                                                   )
    // returns the command name                                    )
    getCommandName().                                              )
    setUserId().                                                   )
    // return the adapter used to format the incoming request      )
    getAdapter(),                                                  )
    .

}
```

Fig. 4C(ii)

```
// processRequest
// This is the main processing unit of the web controller
// It is responsible for the execution of a command within a transaction
//
processRequest(RequestObject req, ResponseOject res) {
    // create a command context object based on the input request
    CommandContext commandContext = createCommandContext(req,res);
    try {
            //
            beginTransaction();
            // set session data in command context
            retrieveSessionData(commandContext);

            // look up and instantiate command to be executed
            ECCommand command = prepareRequest(commandContext);

            // set input properties for command
            command.setRequestProperties(commandContext.
            getRequestProperties());

            // set commandContext for command
            command.setCommandContext();

            // execute command
            command.execute();

            // update session data based on info from command context
            updateSessionData(commandContext);

            // retrieve response properties from command
            responseProperties = command.getResponseProperties();

            // get a response view command
            viewCommand = prepareRespone(responseProperties, commandContext).

            // execute the view command
                    if (viewCommand != null) {
                            viewCommand execute().
                            }
                    commitTransaction()
            } catch (Exception e) {
                    //
                    rollbackTransaction();
                    //
                    handleError(e,commandContext);
                    }
            }
    }
```

Fig. 4C(iii)

```
//
// WebController is the abstract base class the handles any implementation that is
// common for all web controllers
//

abstract class WebController {

    CommandContext createCommandContext(RequestObj req, ResponseObject res) {
            // save request object and response object in command context
            // also extract request parameters, request name, adapter type
    }

    ViewEntry getViewEntry(String commandName, CommandContext commandContext) {
            // look up view based on view name, storeId and device type
    }

    UrlEntry getUrlEntry(String commandName, CommandContext commandContext) {
            // look up url entry based on command name and storeId
    }

    ECCommand instantiateCommand(ViewEntry viewEntry, CommandContext          410
            commandContext) {
            // instantiate command based on interface for view command, store id
    }

    ECCommand instantiateCommand(UrlEntry urlEntry, CommandContext            410
                commandContext) {
            // instantiate command based on command interface, store id
}
```

Fig. 4C(iv)

```
//
// HttpWebController handles any implementation that is specific to the Http protocol
//

public static class HttpWebController {
        ECCommand prepareRequest(CommandContext) throws Exception {
        // look up url entry from URLREG based on name and store id

        UrlEntry urlEntry =getUrlEntry(commandContext.getCommandName(),commandContext);

        if (urlEntry == null) {
                // look up view based on view name, storeId and device type
                ViewEntry viewEntry =
                getViewEntry(commandContext.getCommandName(),commandContext);
                command = instantiateCommand(viewEntry,commandContext);

        } else {
                // check for https redirection
                if (urlEntry.isHttps() && (!commandContext.isHttps()) {

                        ViewEntry viewEntry = getViewEntry("HttpsRedirectView", commandContext);

                        // instantiate command based on interface for view command, store id
                        command = instantiateCommand(viewEntry,commandContext);

                } else {
                        // instantiate command based on command interface, store id
                        command = instantiateCommand(urlEntry,commandContext);
                }
        }
        return command;
}
        //
        // prepareResponse
        //

                ECCommand prepareResponse(TypedProperty responseProperties, CommandContext
                commandContext)throws Exception {
                        // return view command;
                }

                retrieveSessionData(CommandContext commandContext) {
                        // retrieve session data from session context and set it in command context
                }

                updateSessionData(CommandContext commandContext) {
                        // retrieve session data from command context and set it in session context
                }

        }

}
```

Fig. 4C(v)

```
//
// TypedProperty - an extended Hashtable that is used passed request and
// response information to and from a command
//
class TypedProperty extends Hashtable {
    .
    .
    String getString(String parameterName) {
        // return the values of a parameter as a String
    }

    String getInteger(String parameterName) {
        // returns the value of a parameter as an Integer
    }
    String[] getStringArray(String parameterName) {
        // return the values of a parameter as an array of String
    }
    putParameter(String parameterName, Object parameterValue) {
        // store the parameterValue against a parameterName
    }
    .
    .
}
```

FIGURE 5

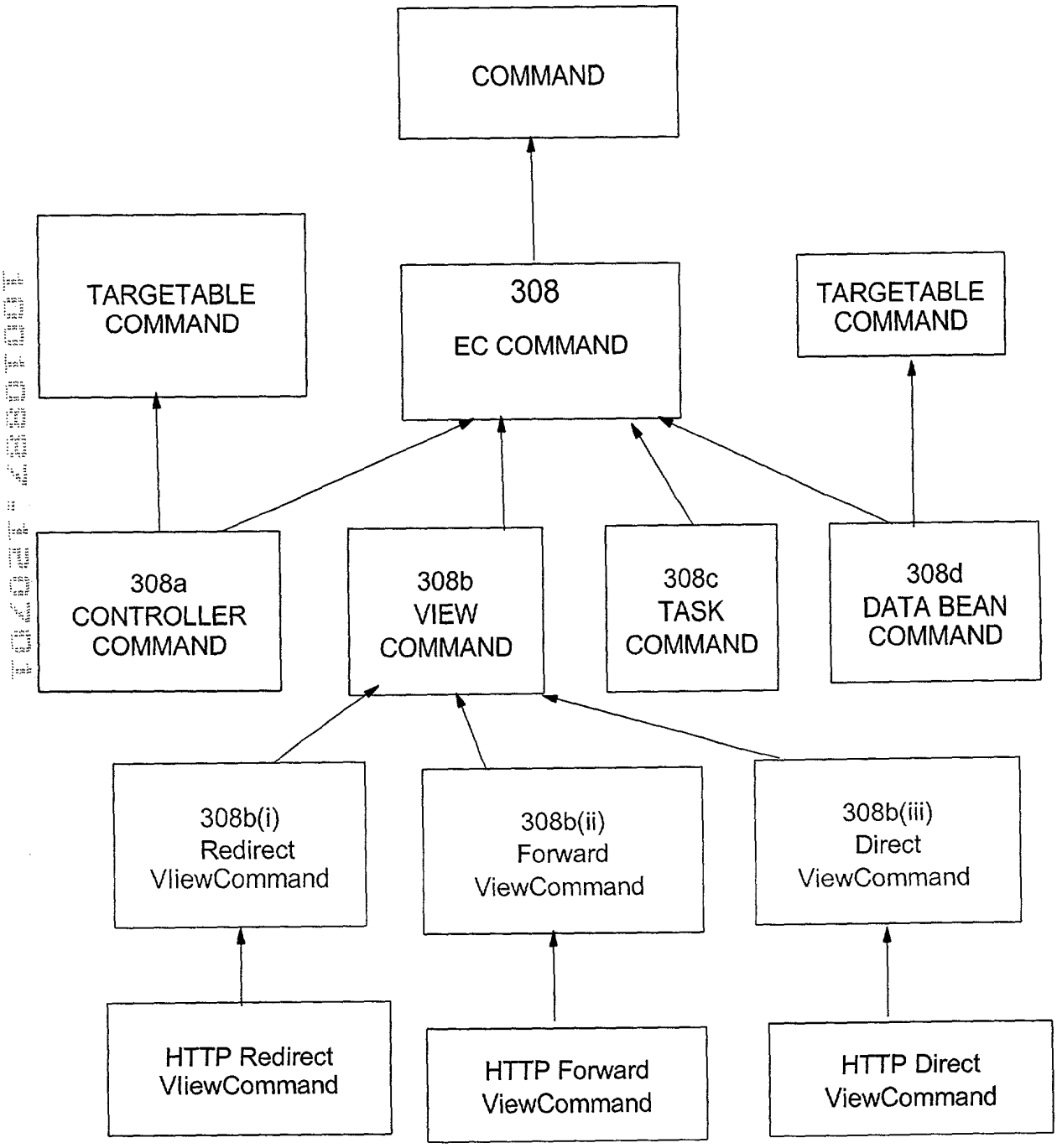

COMMAND
TARGETABLE COMMAND
308
EC COMMAND
TARGETABLE COMMAND
308a
CONTROLLER
COMMAND
308b
VIEW
COMMAND
308c
TASK
COMMAND
308d
DATA BEAN
COMMAND
308b(i)
Redirect
VliewCommand
308b(ii)
Forward
ViewCommand
308b(iii)
Direct
ViewCommand
HTTP Redirect
VliewCommand
HTTP Forward
ViewCommand
HTTP Direct
ViewCommand

Fig. 6A

```
//
//The MQ Adapter listens for incoming messages from the network
//
public class MQSerialAdapter {
        .
        //
        serviceLoop() {
                while (true) {
                        // get a message from the queue
                        MQMessage msg = getRequestFromQueue();

                        // create an adapter to transform and process this message
                        XmlAdapter adapter = createAdapter(MQMessage msg);
                        adapter.processRequest();
                }
        }
        .
        .
}

public class XmlAdapter implements DeviceFormatAdapter {

        processRequest() {
                // convert from message from xml format into a
                // RequestObject
                RequestObject reqobj = createRequestObject(job);
                //
                XMLWebController.processRequest(reqobj, job);
        }

}
```

Fig. 6B

```
//
// XMLWebController handles any implementation that is specific to the MQAdapter
//

public class XMLWebCobtroller extends WebController {
        ECCommand prepareRequest(CommandContext commandContext) throws Exception {

                // look up url entry from URLREG based on command name and storeId
                UrlEntry urlEntry = getUrlEntry(getCommandName(),commandContext);

                // instantiate command based on command interface, store id
                command = instantiateCommand(urlEntry,commandContext);

                //
                return command;
        }

        ECCommand prepareResponse() throws Exception {
                // MQ don't need any response view
                return null;
        }

        retrieveSessionData(CommandContext commandContext) {
                // noop - MQ don't have session info
        }

        updateSessionData(CommandContext commandContext) {
                // noop - MQ don't have session info
        }

}
```

Fig. 7A

```
//
// Scheduler
// The scheduler runs background jobs. They can be jobs that is to be executed only
// once at a specified time or can be jobs that are to be run at regular intervals.
// Jobs are added to the database with the request information, a preferred start time, user id
// and or frequency intervals.
// Job can be added from the browser or from another command
//

public class Scheduler {
    //
    serviceLoop() {
            while (true) {
                // sleep time is determined by the start time of next job
                sleepUntilNextJobIsToBeRun();

                // retrieve the job that need to be executed now from the
                // database
                SchedulerJob job = getReadyToRunJob();

                // allocate a thread to run the job
                SchedulerThread thread = getThreadToRunJob(job);

                // start the thread
                thread.start();
            }
    }
}
```

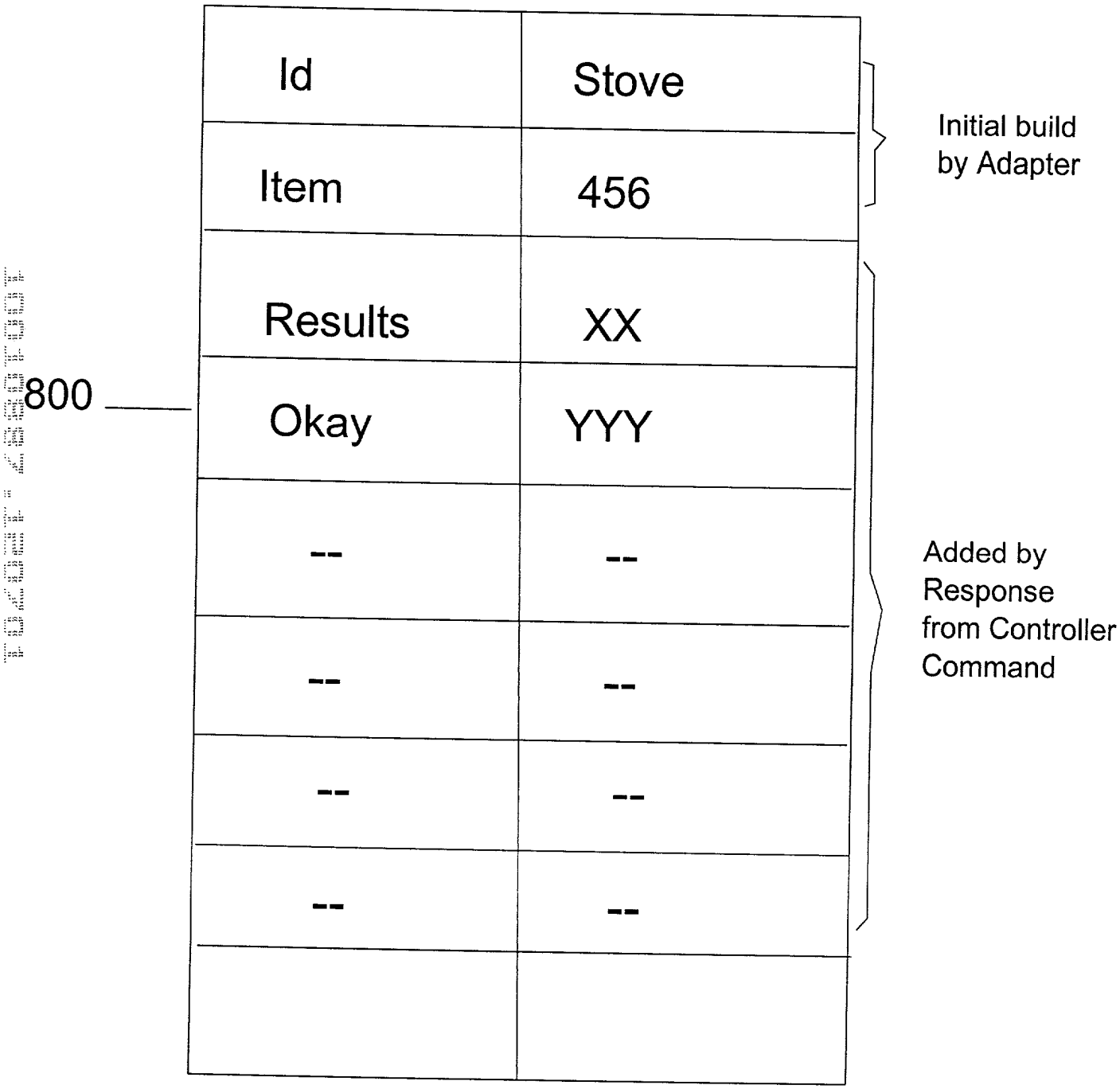
Fig. 7B

```
//
// SchedulerThread
//
public class SchedulerThread {
        SchedulerJob job;
        run() {
            service(job);
        }
        service(SchedulerJob job) {
            // create a scheduler adapter to process the job
            SchedulerAdapter adapter = createSchedulerAdapter(job);
            adapter.processRequest();
        }
}


//
// SchedulerAdapter
// The scheduler adapter is responsible for converting a scheduler job into a request object
// and pass on to the SchedulerWebController
//
public class Scheduler Adapter implements DeviceFormatAdapter{
    processRequest () {
            // convert from scheduler job info into a RequestObject
            RequestObject reqobj = createRequestObject(job);

            // pass request to SchedulerWebController to process
            SchedulerWebCobtroller.processRequest(reqobj, job);
    }
}
```

Fig. 7C

```
//
// SchedulerWebController handles any implementation that is specific to the scheduler
//

public static class SchedulerWebCobtroller extends WebController {
        ECCommand prepareRequest(CommandContext commandContext) throws Exception {

                // look up url entry from URLREG based on command name and storeId
                UrlEntry urlEntry =
                getUrlEntry(getCommandName(),commandContext);

                // instantiate command based on command interface, store id
                command = instantiateCommand(urlEntry,commandContext);

                //
                updateDatabase("jobStarted");

                return command;
        }

        ECCommand prepareResponse() throws Exception {

                // update scheduler database
                updateDatabase("jobCompleted");
                // a background job do not return a view
                return null,
        }

        retrieveSessionData(CommandContext commandContext) {
                // noop - scheduler don't have session info
        }

        updateSessionData(CommandContext commandContext) {
                // noop - scheduler don't have session info
        }

}
```

FIGURE 8


800
Id
Stove
Item
456
Results
XX
Okay
YYY
--
--
--
--
--
--
--
--
Initial build by Adapter
Added by Response from Controller Command

906

Fig. 9

	ViewName	StoreId	DeviceType	Interface Name	ClassName	Properties
1	A	0	Browser	HttpForward ViewCmd	HttpForward ViewCmd Impl	docname= a.jsp
2	A	1	Browser	HttpForward ViewCmd	HttpForward ViewCmd Impl	docname= a1.jsp
3	A	2	Browser	HttpForward ViewCmd	HttpForward ViewCmd Impl	docname= a2.jsp
4	A	3	Browser	HttpForward ViewCmd	HttpForward ViewCmd Impl	docname= a1.jsp
5	A	0	PVCDevice	HttpForward ViewCmd	PVCForward ViewCmd Impl	docname= a.jsp
6	A	1	PVCDevice	HttpForward ViewCmd	PVCForward ViewCmd Impl	docname= a1.jsp
7	A	2	PVCDevice	HttpForward ViewCmd	PVCForward ViewCmd Impl	docname= a2.jsp

902

904

900